1

ARI Research Note 88-06

Models of Incremental Concept Formation

John H. Gennari, Pat Langley, Doug Fisher
University of California at Irvine

for

Contracting Officer's Representative
Judith Orasanu

BASIC RESEARCH LABORATORY
Michael Kaplan, Director

U. S. Army

Research Institute for the Behavioral and Social Sciences

February 1988

U. S. ARMY RESEARCH INSTITUTE
FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency under the Jurisdiction of the

Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON **Technical Director**	**WM. DARRYL HENDERSON** **COL, IN** **Commanding**

Research accomplished under contract
for the Department of the Army

University of California at Irvine

Technical review by

Dan Ragland

Accesion For	
NTIS CRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	

Dist	Avail and / or Special
A-1	

COPY INSPECTED 6

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ARI Research Note 88-06	2. GOVT ACCESSION NO. ADA191598	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Models of Incremental Concept Formation		5. TYPE OF REPORT & PERIOD COVERED Interim Report March 1986-March 1987
		6. PERFORMING ORG. REPORT NUMBER --
7. AUTHOR(s) John H. Gennari, Pat Langley, and Doug Fisher		8. CONTRACT OR GRANT NUMBER(s) MDA90385C0324INT
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of California at Irvine Department of Information and Computer Science Irvine, CA 92715		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 20161102B74F
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Institute Office of Basic Research Alexandria, VA 22333-5600		12. REPORT DATE February 1988
		13. NUMBER OF PAGES 32
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) U.S. Army Research Institute for the Behavioral and Social Sciences, 5001 Eisenhower Avenue, Alexandria, VA 22333-5600		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE --

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

--

18. SUPPLEMENTARY NOTES

--

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Cognitive Psychology	Recognition
Concept Formation	Learning
Tutors	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This ~~research note~~ document reviews three previous models of incremental concept formation and presents a model, CLASSIT, that extends these earlier systems. All of the models integrate the process of recognition and learning, and all can be viewed as carrying out a search through the space of possible concept hierarchies. In an attempt to show that CLASSIT is a robust model of concept formation, empirical studies of the system's behavior under a variety of conditions are also presented. (Keywords: co[illegible] [illegible]

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

1. Introduction

Many aspects of human learning can be modeled as a gradual process of *concept formation*. In this view, the agent observes a succession of objects or instances from which he induces a set of concepts that summarize and organize his experiences. This model differs from the simpler task of 'learning from examples' along two dimensions. First, the concept formation process does not rely on a tutor to presort objects into alternative classes. Second, in concept formation the agent must acquire a *hierarchy* of concept descriptions, rather than the single level that is usually assumed in learning from examples.

Research on conceptual clustering (Michalski & Stepp, 1983; Fisher & Langley, 1986) addresses both of these issues, but existing clustering systems require all data at the outset. We believe that much human concept learning is *incremental* and that models of this process should reflect that fact. We will reserve the term *concept formation* for incremental methods of conceptual clustering. In addition to modeling human behavior, there are practical reasons for preferring incremental systems to nonincremental ones: they let one efficiently incorporate new instances into memory as they are encountered; and they have the potential for responding to changing environments or *concept drift* (Schlimmer & Granger, 1986).

Concept formation can be viewed as a mechanism for automatically organizing memory. It accepts experiences as input – either directly from the perceptual system or after they have been processed by other modules – and it stores them in a conceptual hierarchy, modifying that hierarchy and its component concepts at the same time. Thus, concept formation is intimately linked to recognition, though the products of concept formation can be used in other aspects of intelligent behavior such as problem solving and language understanding.

Below we examine the concept formation task from two different viewpoints – in terms of conceptual hierarchies and in terms of heuristic search. With these metaphors as background, we then examine three earlier systems that have modeled this process – EPAM, UNIMEM and COBWEB. After this we introduce CLASSIT, a new model of concept formation that borrows ideas from previous work but which also improves upon it. We include a simple example to clarify CLASSIT's operation and then present some empirical studies of the system's behavior.

2. An Overview of Incremental Concept Formation

There are four characterizing aspects of concept formation: the nature of the instance descriptions, the nature and organization of the concept descriptions produced, the performance system used in recognition, and the mechanisms used to acquire concepts. As we describe CLASSIT and its predecessors, we will see how they differ along these four dimensions. However, they also have much in common; below we attempt to characterize the underlying approach to concept formation that these systems share. We attempt this from two perspectives – by examining the relation between learning and the performance component and by viewing the concept formation process in terms of heuristic search.

2.1 Recognition and Learning

Concepts vary in their specificity; a very general concept will cover a variety of instances, whereas a more specific one will cover only a subset of those instances. This partial ordering leads naturally to the notion of a concept *hierarchy*, in which specific concepts are stored below more general concepts of which they are a special case.[1]For example, the concepts for *collie* and *dachshund* are specializations of the more general *dog* concept, and thus would occur below the latter in a concept hierarchy. Similarly, the concepts for *dog* and *cat* are specializations of the class of *quadruped mammals*, and so would occur beneath this node in the hierarchy. All four of the models we will examine use some form of concept hierarchy.

Such a memory organization suggests a natural algorithm that integrates the processes of classification (recognition) and learning. The basic approach is to treat the concept hierarchy as a discrimination network for sorting instances into appropriate categories. This process occurs recursively at each level of the hierarchy, starting with the most general node and moving downward:

- At each level, one must decide into which of the alternative categories the instance should be placed, or whether to create an entirely new category based on that instance.

[1] Readers should be careful to distinguish such a concept hierarchy from the partial ordering on concepts that Mitchell (1982) describes in the context of search for an appropriate concept description. The former is a data structure that exists in memory; the latter is a search space.

- If one selects an an existing category, then the definition or description of that category must be modified to take the new instance into account.
- One then retrieves the subcategories of this node and recursively applies the same process to each of these classes, determining the best category and modifying it in turn.

This process continues until one reaches a terminal node in the concept hierarchy or until one reaches some desired level of specificity. Note that the act of recognition and the incremental modification of concepts are intimately intertwined in this framework. One cannot classify an object without incrementally modifying either the structure of the concept hierarchy (by creating a new disjunct) or the definitions of existing concepts. Similarly, one cannot learn apart from this classification process. All of the systems we will review have this integrated character.

2.2 Concept Formation as Search

Another common aspect of the concept formation models we will examine is that all can be described in the language of heuristic search. As we describe each system, we will characterize its learning process in terms of their operators for generating new states, their evaluation function, and their basic search control scheme.

In each case, the basic search space consists of the very large set of distinct concept hierarchies that can be used to classify the instances. One begins with a degenerate hierarchy containing a single category based on the first instance. As one encounters new instances, learning operators alter the hierarchy by adding subcategories at lower levels, by adding new disjuncts at existing levels, and by modifying the definitions of existing categories. Some models also restructure the hierarchy by removing links, removing entire categories, and combining or splitting categories. Each such modification constitutes a step through the space of possible concept hierarchies in search of an optimal organization of knowledge.

At each point a number of choices present themselves, and the techniques used to select one choice over another can be viewed as evaluation functions for guiding the search in promising directions. Intuitively, these functions measure the *quality* of the alternative concept hierarchies with respect to the instances they summarize. They tell the system when to incorporate an instance into an existing category, when to create an entirely new category (a disjunct), and when to reorganize the hierarchy in order to recover from nonrepresentative data. We will see that the existing models of concept formation differ widely in the evaluation functions they employ, and that this gives their behaviors a quite different flavor despite the underlying similarity.

3. Earlier Research on Concept Formation

In this section we review in some detail three earlier models of incremental concept formation – Feigenbaum's EPAM, Lebowitz's UNIMEM, and Fisher's COBWEB. We will see that each system operates within the same basic framework, but also that each system makes some definite improvements over its predecessors. Taken together, these developments constitute an evolutionary trend that has led directly to our current research on concept formation. In each case, we describe the models in terms of their representation of instances and concepts, their basic learning operations, and their search control and evaluation function.

3.1 Feigenbaum's EPAM

Feigenbaum's EPAM (1963) was the first system to model the process of incremental concept formation. This system was developed as a psychological model; it replicated a number of learning phenomena observed in psychological experiments. EPAM uses instance descriptions consisting of a set of structured attribute-value pairs. A discrimination network is created from these attributes, and some very simple learning operators incrementally modify the concepts produced.

In its most typical domain, instances are simply words – a short sequence of letters. In effect, each position in the sequence is an attribute, with 26 possible attribute values. Although EPAM is capable of using instance descriptions with an arbitrarily complex structure, in this domain, instance descriptions represent a very simple perception of the environment.

The discrimination network in EPAM can be viewed as an early and somewhat degenerate version of the concept hierarchies outlined in the preceding section. Each branch point corresponds to a specific attribute, and instances are classified at that node according to the value of that attribute. As an instance is classified, the system descends through the net, looking at different attributes in turn, until a leaf node is reached. Leaf nodes correspond to concepts; this is the only place in the network where EPAM stores an intensional concept description (which Feigenbaum called an *image*). The concept description is a set of relevant attribute values for that concept. When a leaf is reached, either the instance fails to match the concept and *discrimination* occurs, or the instance is recognized and concept *familiarization* occurs. These are the two operators EPAM uses to search for an optimal concept hierarchy.

The discrimination process creates a new disjunct at a given level in the network. In EPAM, there is never any question about when to perform this operation. Each distinct instance is a member of a distinct concept, so whenever the network tries to put two different instances together, discrimination must occur. When the net successfully classifies an instance, familiarization occurs. This is simply the addition of an additional predictable attribute to the concept description. Although in some sense this makes a concept 'stronger', there is no evaluation function whose value is raised.

The most obvious deficiency of EPAM is the lack of an explicit concept hierarchy. Although the internal nodes of the discrimination network could by used to define higher-level concepts, EPAM only uses the concepts at the leaves: there are no explicit subordinate or super-ordinate concepts. In addition, although one can describe EPAM as searching through a space of possible categories, the evaluation function used is trivial because concepts are so exactly defined. In particular, the choice about which operator to apply is simply 'Does the instance match the category?' Because both the instances and the concepts are described by a list of symbolic values, the answer is always a simple 'yes' or 'no'.

EPAM certainly has many strengths. It was the first system to create concepts incrementally: the discrimination network is created and modified as instances are received. Although simplistic, concept descriptions are a representation of the instances that make up that concept. In order to classify or recognize an instance, only some attributes are inspected: namely those attributes that distinguish the new instance from others in the network.

3.2 Lebowitz's UNIMEM

In contrast to EPAM, Lebowitz's UNIMEM system (1986) uses a general hierarchy of concepts, and is designed to work with more complex instance descriptions. UNIMEM evolved from the idea of Generalization-Based Memory (GBM). Although of obvious importance for an integrated system, the focus of this paper is not memory organization, which we view as belonging to a somewhat separate system. Therefore we shall touch only briefly on GBM as we discuss the concept representation of UNIMEM.

UNIMEM was designed to abstract concepts from a number of complex domains. Instances are described by attribute-value pairs, and attributes may have numeric as well as symbolic domains. For example, one test set consists of descriptions of universities with attributes such as state, private/public control, (symbolic attributes) average SAT scores, expenses, and number of students (numeric attributes). UNIMEM is incremental: it uses concept descriptions that summarize the instances, and it builds and modifies the hierarchy as new instances arrive.

For UNIMEM, the concept descriptions in the hierarchy are called GEN-NODEs; these make up Generalization-Based Memory. A description consists of a list of predicatable attribute values (like EPAM), pointers to lower, more specific GEN-NODEs (if there are any), and a network containing all the instances classified under that node. Finally, there are 'confidence counters' associated with each of the attributes listed as predictable.

As UNIMEM descends through the hierarchy, it uses the attribute list of each node to decide if the instance should be classified under that node. If some number of instance attributes match the attribute list in the concept description, then the instance is placed in that concept and the process continues with the concept's children. The number of attributes necessary for this match is a parameter to the system. Note that in some cases, an instance may be classified in more than one concept.

Eventually a node will be reached where the instance is not placed in any of its existing children. This can happen at any level of the hierarchy, but it is guaranteed to happen at the leaf nodes. At this point, either the instance is used to make a new concept node, or it is added only to the parent concept. A new concept is created only if there is a set of sufficiently similar instances already residing in the parent node.

Table 1

The control structure of UNIMEM

Function UNIMEM:

FIND the most specific concept(s) for the new instance, then, UPDATE each of these concepts.

Function FIND (new-instance, root):

(a) Increase confidence counters for matched attributes in root; decrease counters for those not matched. If confidence is low enough, delete attribute. (If enough attributes are deleted, delete node.)

(b) For each child node with enough attributes that match the new-instance call FIND (new-instance, child).

(c) If there are no child nodes that match, return root. Otherwise, return the union of the recursive calls in step (b).

Function UPDATE (new-instance, node):

(a) If none of the instances stored under node are similar enough to new-instance, then store new instance under node.

(b) Otherwise, create a new child under node with new-instance and the similar old instances. Remove the old instances from node's instance network.

Each time an instance is placed into a concept, the confidence counters are modified by the attribute values of the new instance. If the attributes values in the concept do not match the new instance, then the confidence is decreased. If the confidence falls below a threshold, then that attribute is removed from the list in the concept description (i.e., the attribute is no longer sufficiently predictable). In turn, if enough attributes are deleted, the entire concept is discarded. For a summary of UNIMEM's control structure, see Table 1.

By allowing both numeric and symbolic instance attributes, UNIMEM uses a more general-purpose instance description than EPAM. However, it also increases the complexity of the match process. Once the symbolic domain is abandoned, matching returns a degree of match rather than a simple match/no match result. To find the degree of match between two numeric attribute values, Lebowitz uses a statistical distance metric. This is an important representational issue and we shall return to it in section 4.

If UNIMEM places an instance into more than one concept, then concepts are overlapping: they do not form disjoint partitions over the instances. In cluster analysis (see section 4), this has been called *clumping*. In some domains, overlapping concepts may more accurately describe the data than disjoint partitions. In addition, clumping is used to make the search strategy more flexible. If two alternative classifications are suggested, the system may choose both and then, at some later time, decide to remove one concept or the other. In effect, UNIMEM is performing a sort of beam search; the duplicated instances represent a limited set of alternative concept hierarchies.

As UNIMEM searches, it applies a small set of operators to move through the space of concept hierarchies. These are:

- adding an instance to a concept;
- creating a new concept node;
- removing a concept node.

A set of parameters determine how these operators are used by the system. For example, the threshold number of attributes in the concept description is a parameter that directly affects how often concepts are discarded by the system. Similarly, a parameter involving the number of attributes shared with other instances controls when to formulate a new concept.

As we have stressed, one of the crucial components for a concept formation system is the evaluation function that guides the search for an optimal concept hierarchy. UNIMEM does not have a well-defined, principled evaluation function. Unlike EPAM, there is some measurement of concept quality based on the number of predictable attributes and the confidence counters on those attributes. This constitutes a kind of evaluation function, since it controls when various operators are used. For example, if there are too few attributes, the concept is removed. However, this measurement depends too much on external parameters. The values chosen for these parameters affects the performance of the system significantly, but currently, these values are chosen empirically by the researcher instead of by the system.

Nonetheless, UNIMEM provides an excellent basis for incremental concept formation. It has clear advantages over EPAM along several dimensions. It uses concept descriptions with weighted attribute lists, and these are organized into a concept hierarchy. It uses richer instance descriptions, with numeric as well as symbolic attribute values. These differences in complexity have led to the idea of a principled evaluation function to be used at decision points in the classification process. Although we are not satisfied with the mechanisms used by UNIMEM, the system has provided the basis for our own research.

3.3 COBWEB

Fisher's COBWEB system (1987) has retained much of the underlying design of UNIMEM. COBWEB uses the same data structure, a hierarchy of concepts, and a similar control structure where the concepts are modified incrementally by each instance. Although there are many similarities between these two systems, there are two important differences: the representation of a concept used, and the evaluation of concept quality used to guide the search for an optimal concept hierarchy.

COBWEB's concept hierarchy is similar to UNIMEM's: a tree of concepts ranging from general to specific. However, it extends all the way to the most specific concepts possible; leaf nodes represent individual instances. These concepts divide the instances into mutually disjoint classes, unlike the clumping of instances allowed by UNIMEM. COBWEB also uses a similar representation for instances; these are simple sets of symbolic attribute-value pairs.

Like UNIMEM, concept descriptions are a list of attributes with attached weights. However, the list is an exhaustive one, and the weights are formally interpretable as attribute value probabilities conditioned on class membership. For instance, a property associated with the class of birds is that it flies with high probability – e.g., $P(flies|bird) = 0.95$. These conditional probabilities represent the proportion of class members with a given attribute value. Smith and Medin (1981) have used the term *probabilistic concepts* to refer to concept representations that incorporate conditional probabilities.

The evaluation function used by COBWEB is called *category utility* and was developed by Gluck and Corter (1985). Category utility was originally derived (and validated) as a means of predicting certain effects observed during human classification.[2]Category utility favors classes which maximize the potential to infer information (see Fisher, in press), as well as being a function that maximizes intra-class similarity and inter-class dissimilarity.

For an attribute value pair, $A_i = V_{ij}$, and class, C_k, intra-class similarity is measured in terms of a conditional probability $P(A_i = V_{ij}|C_k)$; the larger this probability, the greater

[2] For example, Rosch (1978) has identified certain concepts as "Basic Level" categories. These categories are characterised by quick retrieval and recognition. Gluck and Corter propose that their Category Utility measure is maximised when the concepts correspond to these basic level categories. Since the basic level categories indicate the 'preferred' categories for humans, we believe that this measure is a good one for guiding concept formation.

the proportion of class members sharing the same value (V_{ij}), and thus the more predictable the value is of class members. Inter-class dissimilarity is measured in terms of $P(C_k|A_i = V_{ij})$; the larger this probability, the fewer objects in other classes share this value, and thus the more predictive the value is of the class.

The conditional probabilities above only indicate the dispositions of individual attribute values, but these primitive measures of value predictability and predictiveness can be combined to give an overall measure of partition quality. Specifically,

$$CU = \sum_k \sum_i \sum_j P(A_i = V_{ij})P(C_k|A_i = V_{ij})P(A_i = V_{ij}|C_k)$$

represents a tradeoff between intra-class similarity $P(A_i = V_{ij}|C_k)$ and inter-class dissimilarity $P(C_k|A_i = V_{ij})$, that has been summed for all classes (k), attributes (i), and values (j). The probability $P(A_i = V_{ij})$ weights the importance of individual values, in essence saying that frequently occurring values are more important than infrequently occurring ones. This function forms the basis of the more comprehensive measure, category utility.

As COBWEB incorporates new instances into the concept hierarchy, it uses the category utility measure to decide between one of four operators:

- classifying the object into an existing class;
- creating a new class (a new disjunct);
- combining classes into a single class; and
- dividing a class into several classes.

As we argued earlier, these operators can be viewed in terms of search through the space of possible concept hierarchies. For COBWEB, category utility is the evaluation function that controls when each of these operators is used.

The first two operators are similar to ones we have seen before. In order to decide between them, the following algorithm is used: The new instance is tentatively placed in each child and category utility is computed for each of these possible partitions. The best of these (the host that results in the highest value of category utility) is then compared to the partition resulting if the new instance is used to create a new singleton class at that level. Again, category utility is used to decide between these two partitions.

Table 2

The control structure of COBWEB

```
FUNCTION Cobweb (Object, Root (of a classification tree))
    (1) Update counts of the Root
    (2) IF Root is a leaf THEN Return the expanded leaf to accommodate the new object
        ELSE  Find that child of Root which best hosts Object and perform
                  one of the following:
              (a) Consider creating a new class and do so if appropriate
              (b) Consider node merging and do so if appropriate and recursively
                  call Cobweb (Object, Merged node)
              (c) Consider node splitting and do so if appropriate and recursively
                  call Cobweb (Object, Root)
              IF none of the above (a,b, or c) were performed then recursively
                  call Cobweb (Object, Best child of Root)
```

While these first two operators are effective in many cases, by themselves they are very sensitive to initial input ordering. That is, different concept hierarchies will be created from the same data if the initial set of instances are skewed rather than representative of the entire set. The next two operators are designed to adjust a concept hierarchy when the initial observations prove unrepresentative.

These two operators are node *merging* and node *splitting*. The function of merging is to take two nodes from a partition (of n nodes) and 'combine' them in hopes that the resultant partition (of $n-1$ nodes) is of better quality. The two original nodes are made children of the newly created node. Although merging could be attempted on all possible node pairs every time an object is observed, such a strategy would be unnecessarily costly. Instead, when COBWEB incorporates an object, it only considers merging the two best hosts (as indicated by category utility). Likewise, node splitting may also serve to increase partition quality. A node of a partition (of n nodes) may be deleted and its children promoted, resulting in a partition of $n+m-1$ nodes, where the deleted node had m children. Splitting is considered only for the children of the best host among the existing categories. A summary of the control of the four operators used by COBWEB is presented in Table 2.

Node merging and splitting represent inverse operators and allow COBWEB to move bidirectionally through a space of possible hierarchies. In general, node merging is invoked when initial observations suggest that the environment is a space of highly similar objects, relative to the actual structure of the environment. Node splitting is invoked when the environment is more 'compressed' than suggested by initial input. Merging and splitting decrease the sensitivity of COBWEB to input ordering due to their inverse relation to one another.

4. Modeling the Formation of Object Concepts

With these systems as background, we can turn to CLASSIT, a model of concept formation that attempts to improve upon the earlier work. This model has been most strongly influenced by COBWEB, differing mainly in its representation of instances, its representation of concepts, and its evaluation function. However, CLASSIT uses the same basic operators and the same control strategy that Fisher's system employs. Below we describe the differences between the two systems, along with our motivations for introducing these differences. We also review the two models' learning strategies in terms of the search metaphor.

4.1 Real-Valued Attributes

Although symbolic or nominal attributes occupy an important role in natural language, they are much less useful for describing the physical world. When describing a stick in English, one might say that stick is short or long, but one can also distinguish two sticks that differ only slightly in length. This latter capability suggests that humans' representation of real-world objects can include detailed information about the quantitative features of those objects. A variety of real-world attributes can be described using real numbers, including features such as color, which are usually treated symbolically. Since we are concerned with the formation of physical object concepts, we have designed CLASSIT to accept only real-valued attributes as input.[3]

Later we will present an example of how one can represent physical objects using numeric attributes. Although this approach represents some relational information (such

[3] Statisticians have developed methods for clustering objects described in terms of real-valued attributes; these go by the names of *cluster analysis* and *numerical taxonomy* (Everitt, 1974). Unfortunately, these methods are all nonincremental.

as the adjacency of components) implicitly, it does not really restrict the types of objects that can be described. Furthermore, this form of numeric representation seems a more plausible output from a perception system.

The introduction of real-valued data requires an analogous extension in one's representation of concepts. There are two obvious responses. First, One can divide each numeric attribute into ranges; by 'discretizing' the continuous values, one can retain the symbolic concept representation used in COBWEB. Lebowitz (1985) has taken this approach in one version of UNIMEM. Alternatively, one can represent concepts directly in terms of real-valued attributes.

The CLASSIT model retains COBWEB's notion of a probabilistic representation for concepts, storing a probability distribution with each attribute occurring in a concept. However, instead of storing a probability for each attribute value as COBWEB does, (e.g., $P(small) = 0.3$; $P(large) = 0.7$), our model stores a continuous normal distribution (bell-shaped curve) for each attribute. CLASSIT expresses each distribution in terms of a mean (average) value and a variance. For instance, it might believe that the average length of a dog's tail is 1.1 feet and that its variance is 0.65 feet. Attributes with low variance have narrow, tall distributions; attributes with high variance have wide, shallow distributions. In general, categories lower in the concept hierarchy will have attributes with lower variances, since they represent more specific classes with greater within-group regularity.

4.2 CLASSIT*'s evaluation function*

CLASSIT's use of real-valued attributes in both instances and concepts requires a generalization of COBWEB's evaluation function. We would like some measure that retains the tradeoff between within-group similarity (the ability of a class to predict attributes' values) and between-group differences (the ability of attributes' values to predict a class). We cannot use the conditional probabilities in COBWEB's category utility measure, since these terms are not defined for continuous representations. However, if one assumes a normal distribution, these terms generalize naturally into statistical measures of variance.[4]In

[4] In general, variance is defined as $\frac{\sum_{i=1}^{N}(x_i-\bar{x})^2}{(N-1)}$. Note that this equation as written cannot be computed incrementally; all x_i values need to be present in order to compute the variance. However, a standard transformation where the squared term is expanded and the sum of squares is stored makes variance incremental.

particular, the within-group similarity (W) becomes the variance of the values in the group and the between-group difference (B) becomes the variance of the group averages over a partition:

$$W = \frac{\sum_{j=1}^{J} n_j \sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2}{N - J + 1}$$

$$B = \frac{\sum_{j=1}^{J} n_j (\bar{x}_j - \bar{x})^2}{J}$$

where J is the number of groups, N the number of instances, n_j the number of instances in group j, $\bar{x}_j$ the average of group j, and $\bar{x}$ the average over all groups in the partition. Note that the variance for a group is weighted by the number of instances in that group.

Within-group similarity is actually measured by $1/W$, since as the variance rises, within-group cohesion falls. Finally, these equations measure the variance of a single attribute, so we must sum over all attributes to obtain a complete measure of category quality:[5]

$$Category\ Quality = \sum_{i}^{attributes} \frac{B_i}{W_i}$$

This measure lets CLASSIT find clusterings of instances that maximize within-group similarities and that minimize between-group differences. In order to compute category quality incrementally, each node in the concept hierarchy must contain a statistical summary for each attribute in terms of its mean and variance; as we have seen, this corresponds to the probabilistic concept representation found in COBWEB.

[5] If $n_j = 1$ for all groups in a partition, then $W = 0$, which leads to division by zero. To avoid this problem, we use a minimum value for the sum of squares difference, limiting the perceiving ability of CLASSIT; This limit corresponds to the notion of 'just noticeable differences' in psychophysics – a lower limit greater than zero to our perception ability.

4.3 Search Control: Concept Formation as Hill-Climbing

We believe that a reasonable model of incremental human learning should obey an important constraint: that the learner does not retain competing hypotheses in memory. For concept formation, this means that for a given set of instances one does not carry around alternative concept hierarchies. This constraint severely restricts the search strategies available to a learning system. Breadth-first search and beam search techniques both maintain a frontier of competing hypotheses; depth-first search uses backtracking, implying that a stack of previously considered hypotheses are maintained. However, there is a simple search strategy which obeys this constraint – *hill-climbing* – and both CLASSIT and COBWEB use this strategy to organize their search through the space of possible concept hierarchies.

Although CLASSIT's representation of instances and concepts has forced us to modify COBWEB's evaluation function, it does not require any change in the basic learning operators or in the basic control structure. Thus, CLASSIT includes the same four basic operators as COBWEB – one for incorporating an instance into a existing concept, another for creating a disjunctive concept, a third operator for merging two classes, and a final one for splitting classes. As each new instance is processed, both systems consider all of the available choices, compute the score of the evaluation function in each case, and select that choice giving the highest score.

This is the essence of hill-climbing, since the models occupy only one state at each point and since they have no memory of previous states. Although the merging and splitting operators allow the systems to recover from earlier decisions, they do not perform real backtracking. They can allow the system to return to the same structure as previous hierarchies, but since more instances have been seen, the concept descriptions within the hierarchy are quite different.

We feel that this type of hill-climbing search should be at the heart of any system that claims to model the incremental aspects of human learning. Other search mechanisms make unreasonable assumptions about the nature of human memory. For example, because they use backtracking, depth-first search methods require one to retain in memory a complete trace of the hierarchy's growth. Similarly, breadth-first search schemes require one to maintain an a set of alternative, competing concept hierarchies. The hill-climbing organization used in CLASSIT makes minimal demands on memory while providing a robust framework for incremental learning.

5. A Detailed Example

Now that we have examined CLASSIT's representation, control structure, and evaluation function, let us demonstrate the system's behavior in more detail by stepping through a sample execution. However, we must first consider the the specific input domain that we will use in this example and in our later discussions. It should be made clear that CLASSIT is not restricted to this domain; it can learn any concepts that can be described by a set of real-valued attributes.

5.1 The Input Domain

We have tested CLASSIT with the domain of quadruped mammals specified in terms of simple cylindrical descriptions. In particular, these cylinder descriptions are a simplification of Binford's (1981) generalized cylinders. Marr (1982) has argued that this representation represents a reasonable approximation of the output of a vision system. This representation of objects is also used by the World Modeler's Project, an on-going joint research project with J. G. Carbonell, R. Granger, and D. Kibler.

As discussed earlier, an instance description for CLASSIT consists of a set of attribute-value pairs, where each value is a real number. In this case, each instance is described by eight cylinder components: head, neck, torso, tail, and four legs. In the runs to be described, we generated the data so that each instance fell naturally into one of four basic categories – cats, dogs, horses, and giraffes – depending on the sizes of the cylinders. Figure 1 shows a typical instance for each of these classes.

Each cylinder includes attributes such as height, radius, and location, and we present these to CLASSIT along with the name of the object with which they are associated. Thus, the system receives attributes of the form 'leg1-radius', 'leg2-radius', 'leg1-length', 'leg2-length', and so forth.[6] We typically use nine or ten attributes per cylinder, so that each

[6] By using identically structured instances (each has the same eight components) and by giving CLASSIT this association between cylinders and component names, we have sidestepped some important issues. We will return to these in section 7.

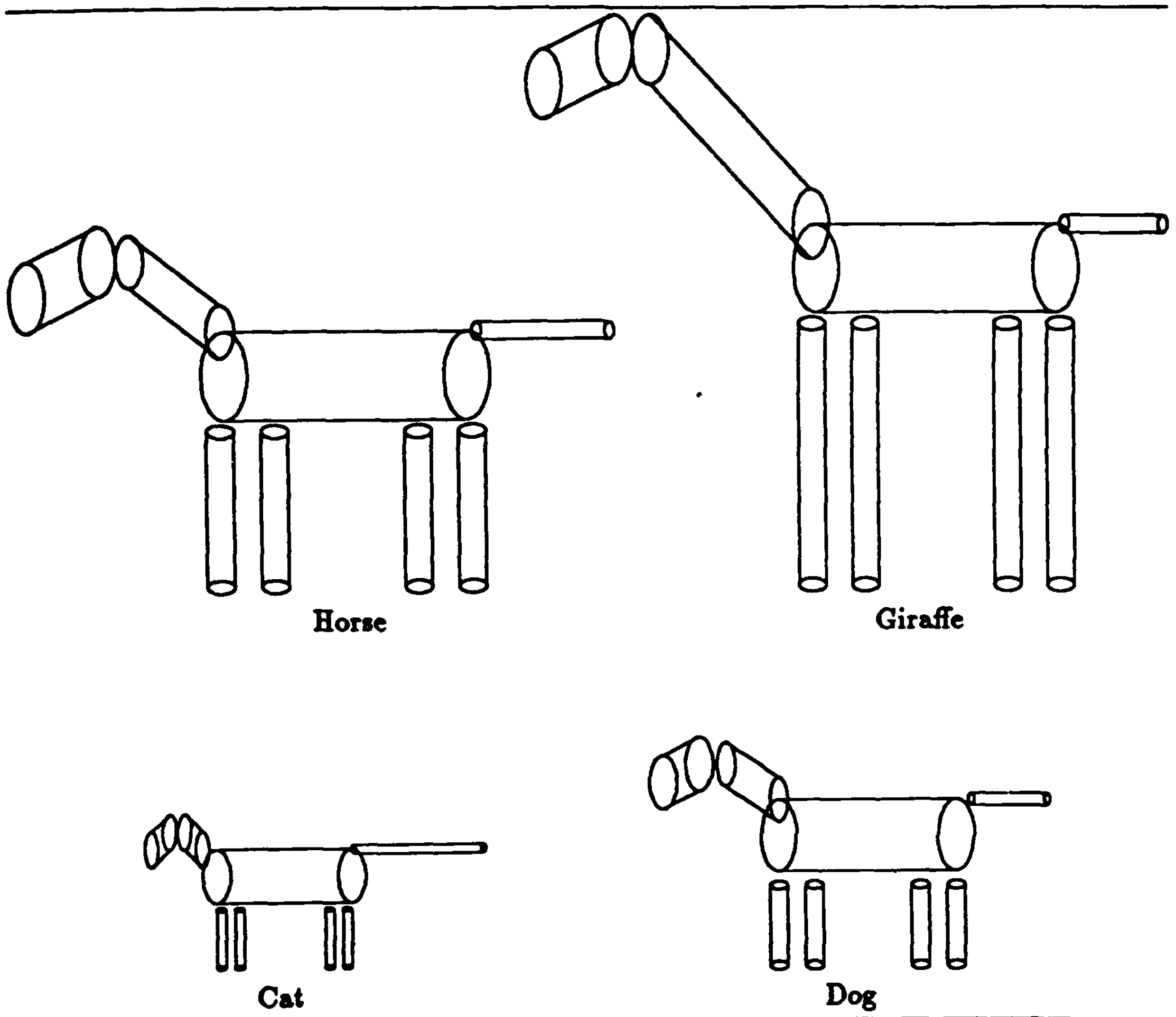
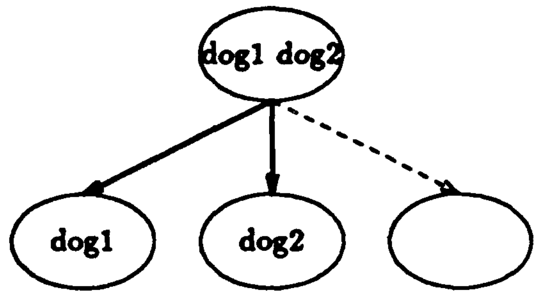


Figure 1. Typical animals.

instances has on the order of 75 attribute-value pairs. We believe that real-world objects have this order of complexity and that a robust concept formation system should be able to handle instances of this form.

In producing data for our test runs, we used a template for each basic category (a 'Platonic form') and constructed instances as variations on these categories using a random number generator. Thus, each instance is a member of one of the four categories; there are no giraffe/dog hybrids. However, some instances diverge more from the ideal form than others. The amount of variation is related to a parameter associated with each attribute; some of these are nearly constant within a category, but others vary widely according to a normal distribution. Of course, CLASSIT will produce a concept hierarchy from any set

New instance: dog3



Category Quality: 2.70 8.34 4.31

Figure 2. The hierarchy as dog3 is classified.

of instances, so it is not necessary to use this kind of carefully generated data. However, using such data greatly simplifies our ability to evaluate the system's performance.

5.2 A Sample Execution

In order to demonstrate CLASSIT's operation, let us now step through a typical execution using the type of input described above. The system begins with an empty concept hierarchy. Suppose the first instance is a dog (call it 'dog1'), though this is not known to the program. This instance is used to create the root node of the hierarchy, creating an initial concept description with means based on the observed values of attributes and with minimum variances. Now suppose we present a second instance that is also a dog (call it 'dog2'). By definition, there is always only one concept at the root level of the hierarchy, so this instance is incorporated into the root concept, generating a new set of means and variances.

However, this instance differs somewhat from the first observation, so CLASSIT forms two categories at the second level of the hierarchy, one based on each instance. This gives the structure shown in dark lines in Figure 2.[7]When a third dog (call it 'dog3') is presented, the system again incorporates it into the root node.

When the system moves to the second level, there are three alternatives for storing dog3. CLASSIT can incorporate the new instance into the same class as dog1, it can add it

[7] Of course, CLASSIT does not use the names of instances during classification; we have included them for the sake of clarity.

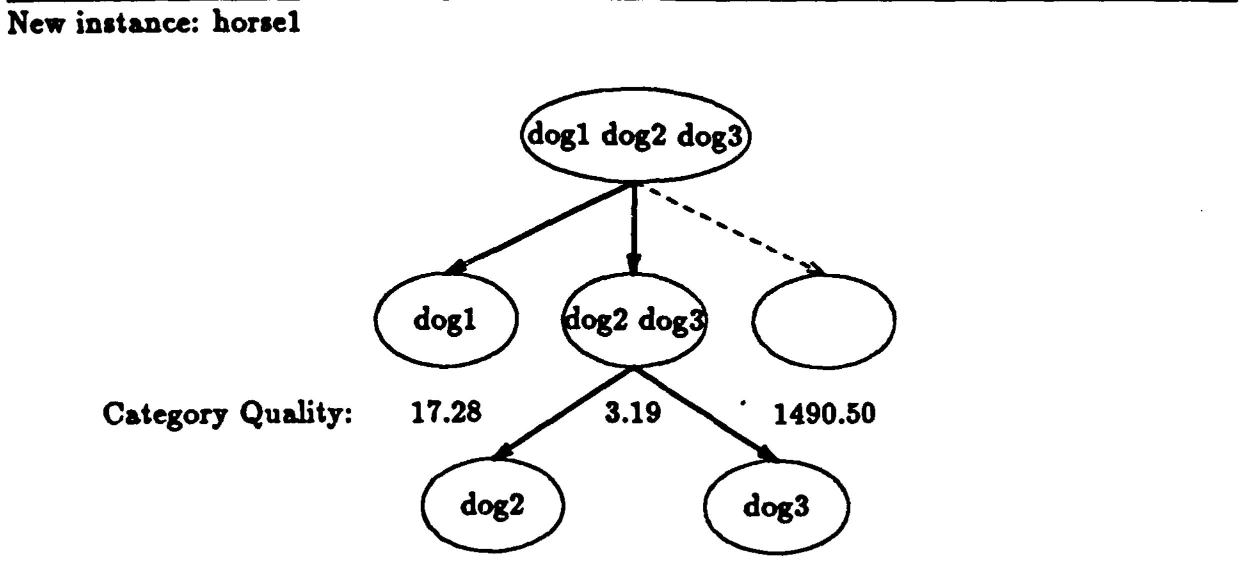


Figure 3. The hierarchy as horse1 is classified.

to the class containing dog2, or it can create an entirely new category. Figure 2 gives the category quality for each of these choices, with the third option (creating a new disjunct) shown with a dotted line. In this case, the highest score results from placing the instance in the same category as dog2; effectively, this is because these two instances are the most similar to each other. In fact, none of the category quality scores are very high, since all of the instances are relatively close. Shortly we will see cases in which greater differences result in much higher scores. We should note that these scores are meaningful only for comparison – their absolute value depends upon the number of attributes in the instances. Figure 3 shows (in dark lines) the hierarchy after dog3 has been incorporated. Notice that, although dog2 and dog3 were placed in the same category, they have led to separate classes at the third level of the hierarchy.

Suppose the next instance is actually a horse. Again the instance is averaged into the root node, and again there are three choices at the second level. Figure 3 presents the category quality scores that would result placing the horse in each of the two existing classes versus creating an entirely new category. In this case there is a very clear preference for making a new disjunct; the horse is quite different from either of the existing classes at this level and this is reflected in the high score associated with creating a disjunct.

Finally, suppose the next three instances are horses and giraffes; Figure 4 presents the hierarchy after these observations have been processed. When a fourth dog is presented,

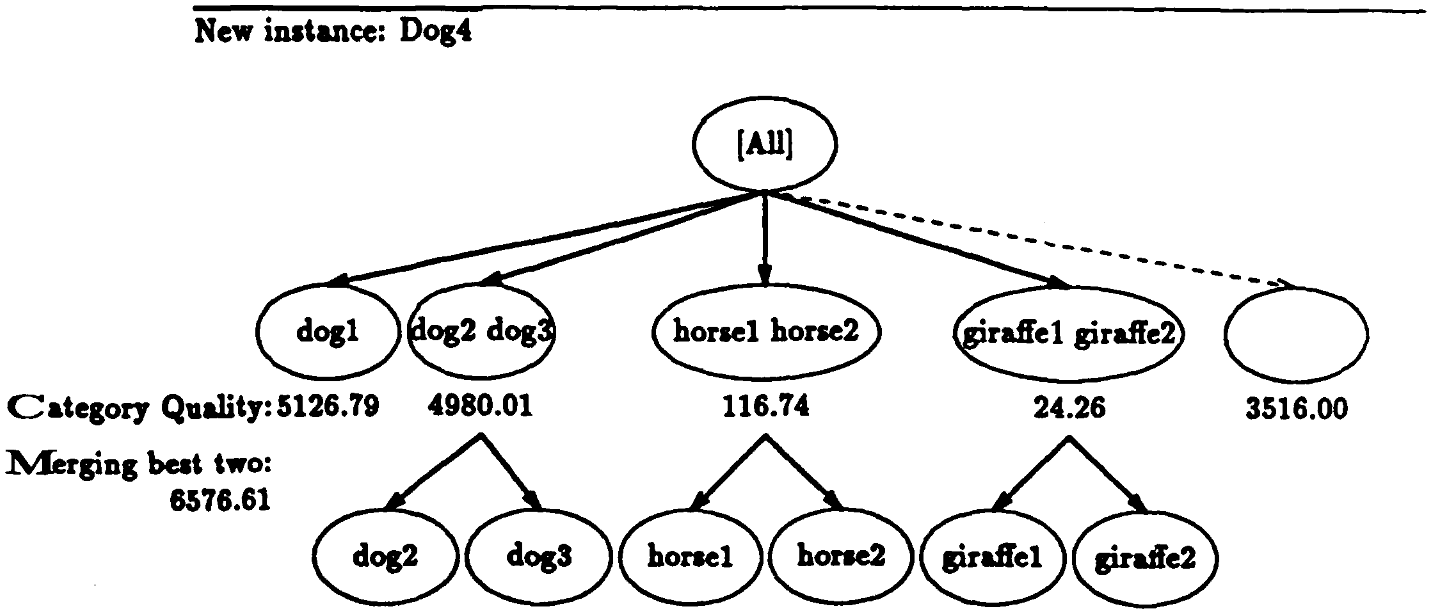


Figure 4. The hierarchy as dog4 is classified.

CLASSIT performs a merge operation. Whenever there are more than two groups in a partition, the system considers merging the best of the two groups. In this case, the two dog groups have the highest category quality. If these two were merged, the resulting partition would have a higher category quality than any of the other options, so CLASSIT selects this action. Figure 5 shows the hierarchy that results from this combination. Note that the two merged concepts still exist, but they have been moved down a level in the hierarchy. After merging, the system had to decide how to add dog4 to level 3 – in this case, a new disjunct was made.

CLASSIT does not halt at this point. The system continues to process new instances, incrementally modifying both its concept descriptions and the concept hierarchy as it encounters new data. Unlike some incremental learning systems – such as Mitchell's (1982) version space method – CLASSIT never achieves a final knowledge state; the system continues to learn as long as new instances are available. Contrary to being a weakness, we believe this is a strength of the model. Human learners behave in this fashion in many domains, and we suspect there are good reasons for this strategy. One of them involves the possibility of concept drift; if the environment has the potential for changing over time, the learner must continue to modify his conceptual structures in response to new data.

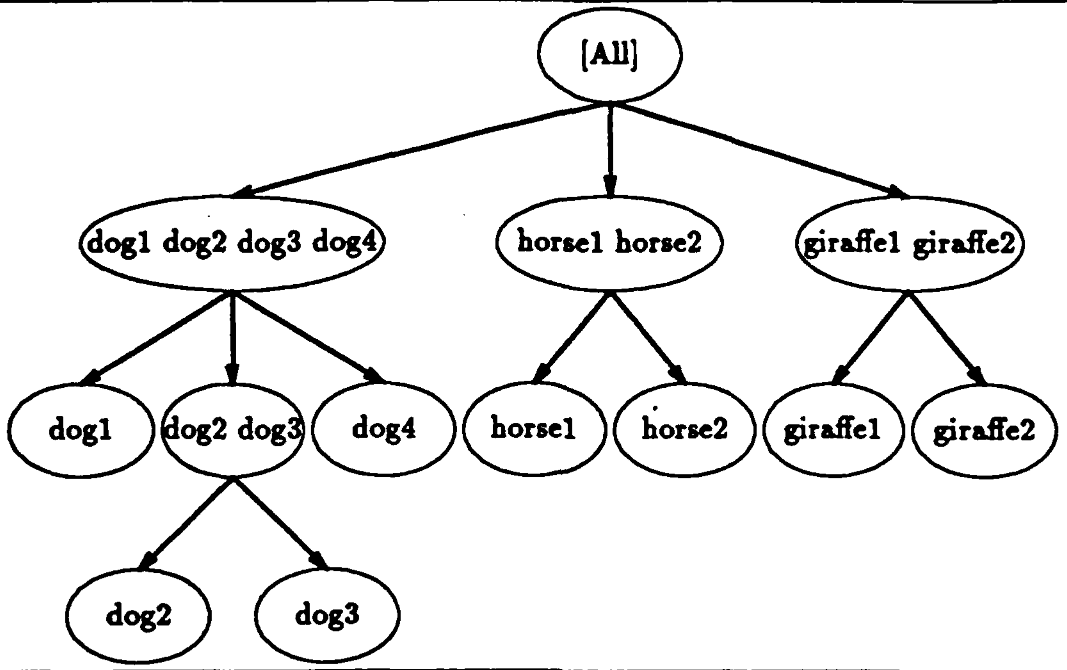


Figure 5. The hierarchy after classifying dog4.

Although we have not tested CLASSIT under such conditions, we expect it would perform in a reasonable manner.

6. Experimental Results

In order to evaluate any AI system, one must test that system under a variety of conditions and measure its performance. In this section, we present some experimental results that demonstrate CLASSIT's robustness along three dimensions: within-group variation, the class/subclass distinction, and memory limitations. Each of the tests involve the same basic domain of quadrupeds that we described in the previous section. We conclude by suggesting some areas for future testing.

6.1 Within-group Variation

Members of a given category vary to different degrees. At one extreme, the variation is zero; every member of a class is exactly the same. In such a case, one would expect a concept formation system to be able to find the different classes very easily. This is the case with CLASSIT. With zero intra-class variation, the $1/W$ term in the category quality equation rises very quickly and the system converges on the optimal concept hierarchy.

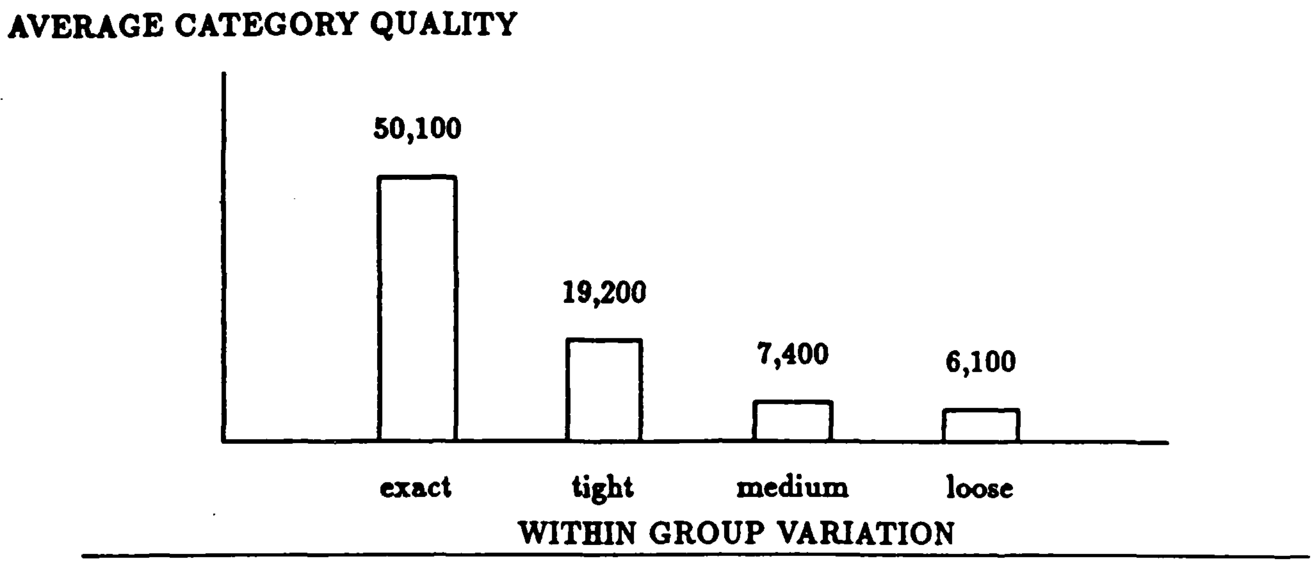


Figure 6. Category quality vs. within-group variation.

At the other extreme, the members of a class may differ greatly from the norm of that class. This should make it more difficult to find the optimal classes. In the worst case, there may be so much within-group variation that classes are formed randomly. This also holds for CLASSIT; the system is unlikely to find meaningful classes in such extreme cases.

Figure 6 summarizes the results from an experiment in which we 'defined' four classes with varying degrees of regularity. The 'exact' condition represents the results of runs on data with zero variation. The 'tight', 'medium', and 'loose' conditions represent data sets in which we introduced more and more variance. The scores for each experiment were averaged over 10 CLASSIT executions, each involving a run on 30 instances. Each score indicates the category quality of the final hierarchy after the 30th instance was processed.[8]

The graph shows that as the regularity within each category decreases, the category quality also decreases. It is difficult to determine whether CLASSIT is actually finding the optimal clustering in each case, since this would require an exhaustive search of the clustering space. However, we have noted that as variation increases, the system's hierarchies tend to diverge from the 'desired' hierarchies used to generate the data. Typically, with greater variation, more than four top-level categories would be identified. This suggests

[8] Since instances are created with a random number generator, different executions within a trial can produce quite different hierarchies.

that CLASSIT's behavior degrades gracefully as the within-group variation increases. The system performs well in orderly environments, but its ability falls off gradually as more variation occurs.

6.2 Discovering Sub-classes

As we noted earlier, concept formation involves more than determining a set of categories – it also requires the construction of a concept *hierarchy*. This mean that a robust concept formation system must be able not only the main classes, but also subclasses and subsubclasses. It must note not only the major similarities and differences that define the basic level categories, but also the smaller regularities and variations that define subcategories. A model of concept formation should place beagles and terriers in different classes, just as it generates different categories for giraffes and dogs.

In order to test this ability, we modified the dog category described earlier so that members of the dog category always fit into one of two subspecies: long-tailed and short-tailed dogs. Given these data, CLASSIT successfully formulated two subcategories of the dog class in 8 out of 10 runs, sorting the expected instances to the correct classes. In the remaining two runs, the system formed three subclasses – one containing short-tailed dogs and two containing varieties of long-tailed dogs. These results are preliminary and we need additional tests involving multiple features and multiple levels, but they are encouraging nonetheless.

Note that the issue of forming subcategories should interact with the issue of within-group variation. As one moves down in the concept hierarchy, the categories represent more specific concepts that, presumably, have less variation among their members. Thus, one might expect CLASSIT's clustering ability to improve as it considers lower levels of the hierarchy. However, the between group variance at lower levels also decreases, since siblings will be more closely related. Whether the system's behavior improves or degrades at lower levels is an empirical question to be anwered by future testing.

6.3 Memory restrictions

Although human long-term memory appears to be effectively unlimited, it seems implausible that one stores every object ever encountered. Yet earlier models of concept formation, in particular UNIMEM and COBWEB, retain complete descriptions of every instance, despite the fact that these descriptions are not needed for the classification process. At any given partition in the hierarchy, only the summary statistics that describe a concept are used to classify an instance.

Like the COBWEB system, CLASSIT was originally designed to continue down the hierarchy until it reached individual instances descriptions. However, we have also tested a version of the system that stops classifying beyond a certain level. This can be thought of as imposing a simple memory limit on the system. Usually this version will produce the same concepts above the cut-off level as the unrestricted system. This is because CLASSIT rarely uses information stored below its current level. The only operation that uses lower levels is the split operation, since this promotes children nodes. If this operation occurs at the cut-off level, then the cut-off and unrestricted versions will produce different hierarchies.

Although it is possible that a series of splits could cascade this difference up to the top levels, in practice it rarely happens. In an experiment on 10 'loose' data sets, 6 produced identical concept hierarchies, 3 hierarchies were different at the cut-off level (in this case, level 4), and only 1 execution produced a hierarchy where level 3 was altered. Presumably the chances of the splits cascading up to level 2 would be quite small. Because the cut-off version is faster than the unrestricted system, we have used it for many of the tests described above.

6.4 Further Tests

There are at least two additional aspects of the input data that should affect CLASSIT's behavior. One of these involves the order in which the instances are presented to the system; the other involves the number of attributes that are actually relevant in distinguishing between categories. Although we have kept these factors constant in the experiments described above, we plan to vary them in future studies.

If an incremental concept formation system is guaranteed to find the optimal concept hierarchy, then the order of the input will not affect its output. However, this is precisely what a hill-climbing strategy (like that used in CLASSIT) does *not* guarantee. Given one ordering of the instances, the system may be fortunate enough to find the optimal hierarchy, but given another ordering, it may become stuck at a local maximum in the hierarchy space. Given a third ordering, the system may get stuck at yet another local maximum. Preliminary experiments show that CLASSIT is somewhat sensitive to the order of input, though the merge and split operators help offset this sensitivity. However, the exact nature of the dependency is not clear. For example, one would expect the system to perform well if it were given all instances of one class, followed by all instances of another, etc., and to perform poorly if the instances of various classes were interleaved. Obviously, we need more rigorous tests to determine the exact nature of the relationship.

The number of irrelevant attributes should also affect CLASSIT's learning behavior. For example, within the domain of quadruped mammals the position of the tail has the same variance and mean for all four templates (horse, giraffe, cat and dog). Although it can vary considerably, this attribute is therefore almost completely useless in distinguishing between different categories. Since CLASSIT sums over all attributes, it effectively ignores this type of useless attribute; when two summations are compared, only those terms that differ are important. In the runs described above, we kept the number of irrelevant features at a constant level: 24 of the 72 total attributes, or 33 percent. However, it is not clear how the system would behave if we greatly increased or decreased the number of such attributes; this is another issue that should be examined in future tests.

7. Directions for Future Research

We believe that CLASSIT constitutes a plausible initial model of human concept formation and that the system incorporates a number of advances of earlier models. However, the existing model has a number of limitations that should be remedied in future efforts, and we discuss these below. Most of them deal with making CLASSIT's input a more plausible model of the output that might be generated by the human perceptual system.

One problem is that CLASSIT is effectively provided with correspondences between the various components in instances and concepts. For example, the system is told that the right front leg in an instance corresponds to the right front leg in the *dog* concept; there is no chance the system will confuse the leg in the instance with the tail in the concept description, or even with a different leg.

Future versions of the model should establish these correspondences for themselves, even at the expense of occasional errors. A brute force approach would first compute all possible mappings between the cylinders in an instance and the cylinders in each concept description, and then determine the optimal match using category quality. For instances containing eight components, this would be expensive but manageable (8! = 40320). This scheme would ensure an optimal match, but we prefer a more heuristic approach that requires less memory and less computation. Using the variances for each attribute in the concept description, CLASSIT would find a match for that cylinder with the least associated variation. Using this as a constraint, the system would then find a match for the next most constrained component and so forth, continuing this process until all components in the concept description had been matched against components in the instance. This 'greedy' approach is not ensured of finding the best match, but it is likely to find an acceptable one with minimal cost.

It is possible to conceive a system where this greedy matching algorithm is used as a pre-processing step to the classification and concept learning processes – after a best match is found, the component labels could be generated and used as input to the existing system. However, such an architecture would contain much duplicated effort. We believe that somehow, the two searches must be occurring simultaneously. That is, as the best match is determined, the object is classified and concept learning occurs. Integrating these two processes remains an open problem.

A second problem is that CLASSIT lacks mechanisms for working with hierarchical objects. Marr (1982) proposes that humans have the ability to describe physical objects at differing levels of aggregation. Thus, a dog might be viewed as a single cylinder at one level; at a lower level, this cylinder is decomposed into eight connected cylinders for torso, neck, head, tail, and legs; at a still lower level, the leg is decomposed into three cylinders corresponding to the thigh, calf, and foot. In our experiments, we presented CLASSIT with only the second of these levels. Future versions of the model should be extended to handle multiple levels of aggregation and to employ these levels intelligently during the match process. This may also help in comparing objects with different structure, since they may be comparable at one level but not another.

A third limitation of the current model is its inability to deal with incomplete objects. If we assume that CLASSIT's input is generated by a vision system, then sometimes component objects will be omitted from an instance description because they are not visible. This means the process of finding correspondences between objects, described above, must be extended to handle partial mappings between instances and concepts.[9] Finding an optimal partial match leads to even greater combinatorial explosion than finding an optimal partial match, but again we will not require CLASSIT to find the best of all possible mappings. In addition, the introduction of hierarchical descriptions will let us keep the number of objects at any given level down to a manageable size, and this will help keep the search within reasonable bounds.

A final extension would introduce selective attention during recognition and learning. The current system inspects every attribute during the classification process, even those that are irrelevant. This may seem unreasonable, but without at least an initial concept hierarchy, there is no way to know which attributes are important and which are not. However, as CLASSIT gains experience with the current domain, it collects statistics that reflect the relative importance of the various attributes. These measures can be used as a focusing device, suggesting which attributes to inspect and which to ignore. In determining which category best matches a new instance, the system need only look at those attributes that are most important for distinguishing between the competing classes. If these attributes are sufficient to make a clear decision, then the other attributes need never be inspected. We believe this would make CLASSIT a more plausible model of the human recognition process.

8. Summary

In this paper, we have tried to develop a unifying framework from which to view incremental concept formation. This framework identifies two issues: we are interested in the learning process as a search algorithm, and we are interested in the representational space used by that search. We have tried to characterize the representation used in both the domain and range of the system, i.e., both the instance descriptions used as input, and the concept descriptions produced as output. Viewing learning as search has directed us

[9] An instance might have missing attributes as well as missing objects, but this matter is much less serious. In such cases, one can simply use the expected (mean) value of the attribute.

Table 3

A summary of concept formation systems

<u>EPAM</u>		
	Input Domain:	Symbolic attribute value pairs, ability to use structured data.
	Output Descriptions:	Images (concept descriptions) stored only at terminal nodes of the hierarchy. An image is a attribute value list.
	Evaluation Function:	Simple match/no match function.
	Learning Operators:	Discriminate (create disjunct) and familiarise (add to a node).
<u>UNIMEM</u>		
	Input Domain:	Numeric and symbolic attribute value pairs.
	Output Descriptions:	Weighted attribute lists organised into a concept hierarchy.
	Evaluation Function:	Parametised, non-uniform evaluation methods.
	Learning Operators:	Create disjunct, add to concept, remove concept.
<u>COBWEB</u>		
	Input Domain:	Symbolic attribute values.
	Output Descriptions:	Probabalistic concept descriptions in a concept hierarchy.
	Evaluation Function:	Category utility.
	Learning Operators:	Create disjunct, add to concept, merge concepts, split concepts.
<u>CLASSIT</u>		
	Input Domain:	Numeric attribute value pairs.
	Output Descriptions:	Statistical concept descriptions in a concept hierarchy.
	Evaluation Function:	Category quality.
	Learning Operators:	Create disjunct, add to concept, merge concepts, split concepts.

to characterize the evaluation function which guides the system through decision points, and the set of operators used to move through the search space.

We have presented four different systems and we have tried to evaluate and compare these systems by using this framework. Table 3 presents a summary of the concept formation systems described in this paper. Each system is characterized by the four points that our framework identifies. From this perspective, it is easy to see how closely the systems are related and that each system makes an improvement over its predecessor. It should be clear that our current research is based on these earlier concept formation systems.

We have argued that an appropriate search for a learning system is one that is characterized by hill-climbing. This means that the operators used should be kept to a minimum and they should not allow arbitrary return to previous hypotheses. We have argued that this search should be guided by a principled evaluation function. We have also suggested that instance descriptions consisting of numeric attribute values provide a reasonable input representation; this representation is plausible as output from a human perception system. Finally, we have presented a structured hierarchy of concept descriptions as the appropriate output for concept formation. CLASSIT is presented as a system which incorporates these ideas. We have tried to give some evidence that this system satisfies these constraints and is reasonably robust under a variety of conditions.

The representation and acquisition of concepts is a complex, interconnected set of problems, and we will not claim to have solved these problems in any absolute sense. However, we believe the basic approach we have described – and reflected in EPAM, UNIMEM, COBWEB, and CLASSIT – constitutes one of the most promising thrusts towards understanding the nature of categorization in computational terms. We encourage other researchers to join in the effort and to construct incremental models of concept formation that extend the initial results that have been achieved to date.

REFERENCES

Binford, T. O. (1971, December). *Visual perception by computer.* Paper presented at the IEEE Conference on Systems and Control, Miami, FL.

Everitt, B. (1974). *Cluster analysis.* Heinemann Educational.

Feigenbaum, E. A. (1963). The simulation of verbal learning behavior. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* New York: McGraw-Hill.

Fisher, D. (in press). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2.*

Fisher, D. & Langley, P. (1986). Conceptual clustering and its relation to numerical taxonomy. In W. A. Gale (Ed.), *Artificial intelligence and Statistics.* Reading, MA: Addison-Wesley.

Gluck, M & Corter, J. (1985). Information, uncertainty and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, 283–287.

Lebowitz, M. (1985). Categorizing numeric information for generalization. *Cognitive Science, 9*, 285–309.

Lebowitz, M. (1986). Concept learning in a rich input domain: generalization based memory. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell, (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Palo Alto, CA: Tioga.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information.* San Francisco: W. H. Freeman.

Michalski, R. S., & Stepp, R. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach.* Palo Alto, CA: Tioga.

Mitchell, T. M. (1982). Generalization as search. *Artificial Intelligence, 18*, 203–226.

Rosch, E. (1978). The principles of categorization. In E. Rosch & B. B. Lloyd (Eds.), *Cognition and categorization*(pp. 27–48). Hillsdale, NJ: Lawrence Erlbaum.

Schlimmer, J. & Granger, R. (1986). Beyond incremental processing: tracking concept drift. *Proceedings of the National Conference on Artificial Intelligence.*

Smith, E. E., & Medin, D.L. (1981). *Categories and concepts.* Cambridge, MA: Harvard University Press.